维普资讯 http://www.cqvip.com

TP312

时态逻辑与程序设计语言

刘 清（江西大学）

摘要

The paper describes that temporal logic can be used as a programming language. It can apply to describe the algorithm for solving problems. It can give the answer after directly executed on the computer.

一、概 述

按照人们的常规，程序设计语言是指计算机能直接执行的机器语言、汇编语言以及其它高级语言，用这些语言描述问题求解的算法，机器能直接执行，并给出问题的解答。

近年来计算机领域里提出的时态逻辑是具有强时态语义的语言，它也可让计算机直接执行，因此，也可用它来描述求解问题的算法，其性能远比传统程序设计语言更为优越，因为用这种语言写的程序，其正确性取决于时态逻辑公式在给定的状态序列δ上的有效性。执行程序语句便是操作一时态逻辑公式。如果给出了操作时态逻辑公式的法则，则可计算出程序运行的结果。

这里给出时态逻辑的描述。

1．符号

(1) 真值：true，false；

(2) 联结词：～（否定），→（隐涵），∧（合取），∨（析取），↔（等值），○（下一时刻），◇（终将有一时刻），□（总有一时刻），⇒（直到），＝（等词），if…then…else…（条件词）；

(3) 量词：(∀…)（全称），(∃…)（存在）；

(4) 变量：x，y，z…；

(5) 函数：f^n，其中$n\geqslant 0$，称n一元函数符；

(6) 谓词：p^n，其中$n\geqslant 0$，称n一元谓词符；

(7) 专用谓词：gets，invar，input，output，halt，empty，fin

2．语法

(1) 时态表达式（TE）

·每个常数a是TE；

·每个独应变量x是TE；

·如果e_1，…，e_n是TE，则$f(e_1, \cdots, e_n)$是TE；

·如果A是时态公式，且e_1和e_2是TE，则if A then e_1 else e_2是TE。

(2) 时态逻辑公式（TF）

·true和false是TF；

·如果P是谓词符，且e_1，…，e_n是TE，则$P(e_1, \cdots, e_n)$是TF；

·＝(e_1, e_2)，gets (e_1, e_2)，input $(U_1, \cdots, U_n)$，output $(U_1, \cdots U_n)$，invar(e)，halt (A)和empty均为TF，其中e，e_1，e_2，以及U_1，…，U_n均是TE；

·如果A，B和C是TF，则～A，$A\to B$，$A\wedge B$，$A\vee B$，$A\leftrightarrow B$，○A，□A，◇A，$A \Rightarrow B$以及if A then B else C都是TF；

·如果X是变量且A是TF，则$(\forall X)A$和$(\exists X)A$都是TF；

例如，$X=2\wedge\bigcirc(y=5)$，$\bigcirc(\square(x=3))\vee\sim(\bigcirc x=6)$，$\bigcirc\square(x=3)\wedge\bigcirc\bigcirc y=8$等等都是TF。

3．语义

时态逻辑公式的解释是在状态序列$\delta=\delta_0$，δ_1，…上给出的。结构$I_\delta[e]$表示在δ上，时态表达式e所取的区域D上的值，类似地$I_\delta[A]$表示在δ上公式A所取的逻辑值域{true，false}上的真值，于是时态语义定义如下：

(1) $I_\delta[a]=a$，其中a是常数；

(2) $I_\delta[x]=I_{\delta(0)}[x]$，时态变量$x$在$\delta$上的值等于它在$\delta$的0一截段上的值；

(3) $I_\delta[\bigcirc e]=I_{\delta(1)}[e]$表示TE○$e$在$\delta$上的值等于$e$在$\delta$的1一截段$\delta^{(1)}$上的值；

维普资讯 http://www.cqvip.com

(4) $I_\delta[f(e_1, \cdots, e_n)]=I[f](I_\delta[e_1], \cdots, I_\delta[e_n])$，表示函数$f(e_1, \cdots, e_n)$在$\delta$上的值等于$e_1, \cdots, e_n$分别在$\delta$上的值，并在$I[f]$作用下，产生$D$中的一个值，其中$I[f]$是将$D$中$n$个元映射到$D$中的一个值，即$I[f]: D^n \to D$；

(5) $I_\delta[P(e_1, \cdots, e_n)]=I[P](I_\delta[e_1], \cdots, I_\delta[e_n])$，表示$P(e_1, \cdots, e_n)$在$\delta$上的值等于$e_1, \cdots, e_n$分别在$\delta$上的值并在$I[P]$作用下产生{true, false}上的一个真值，其中$I[P]$是将D中几个元映射到{true, false}中的一个真值，即$I[P]$，$D^n \to$ {true, false}；

(6) $I_\delta[\sim A]=\sim I_\delta[A]$，$I_\delta[A \wedge B]=I_\delta[A] \wedge I_\delta[B]$，$I_\delta[\bigcirc A]=I_{\delta(1)}[A]$，$I_\delta[\Box A]=I_{\delta(i)}[A]$，其中$i \geqslant 0$，$A$和$B$都是TF；

对含有联结词$\vee$，$\to$，$\leftrightarrow$的公式，均可用$\sim$和$\wedge$定义[2]；同样，对$\Diamond$和$\leftrightarrow$联结词的TF，均可用$\bigcirc$和$\Box$定义[2]。下面给出公式的可满足性和普遍有效性的意义。

可满足性：时态逻辑公式A在δ上有$I_\delta[A]=$ture，则称A是δ上可满足的，记成$\delta \vdash A$。

普遍有效性：时态逻辑公式A在任何δ上都有$I_\delta[A]=$true，则称A是普遍有效的，记成$\vdash A$。

4. 几个特殊谓词公式的意义：

(1) empty是零元谓词，称空公式。

$$\text{empty} \triangleq \text{" "};$$

它在δ上是true当且仅当$|\delta|=0$，其中$|\delta|$表示δ上的状态数目，$\triangleq$表示定义为。

(2) gets(e_1, e_2)是二元谓词式，称获得时态逻辑公式。

$$\text{gets}(e_1, e_2) \triangleq \Box(\bigcirc e_1 = e_2),$$

表示从当前时刻的e_1出发，可在任何时刻的下一时刻获得e_2。

(3) invar(e)是一元谓词式，称不变公式。

$$\text{invar}(e) \triangleq \text{gets}(e, e)$$

表示在invar作用下，e保持不变。

(4) halt(A)是一元谓词式，称停止公式。

$$\text{halt}(A) \triangleq \Box(A = \text{true} \vee A = \text{empty} \to \text{end} \vee \text{stop})$$

其中end和stop是传统程序设计语言中的结束语句和停止语句。

意味着当A是真或空时，则立即结束或停止执行。

(5) input$(U_1, \cdots, U_n)$，$n \geqslant 1$，是至少一元谓词式，称输入公式；output$(U_1, \cdots, U_n)$，$n \geqslant 1$，是至少一元谓词式，称输出公式。

二、时态逻辑与程序设计语言

前面已描述了时态逻辑的基本结构，如果把它视为一种程序设计语言，有必要将其和传统的程序设计语言进行比较，并建立其间的关系。

1. 变量

(1) 局部量。我们将时态逻辑中的时态变量[2]视为传统程序设计语言中的程序变量，亦即不断变化的局部量。

(2) 全程量。时态逻辑中的全程变量与传统程序设计语言中的全程量是一致的，亦即它们均看成不随时间变化的固定量。

2. 表达式

(1) 算术型。凡数值常数和数值变量均为算术型。如果e_1和e_2是算术型表达式，则经算术运算后仍为算术型；另外，如果e为算术型表达式，则$\bigcirc e$是具有时态结构的算术型表达式。

(2) 布尔型。布尔常数true和false以及布尔值的变量均为布尔型表达式；用关系符联结起来的式子$e_1=e_2$，$e_1 \geqslant e_2$等等是布尔型的布尔表达式；如果b_1和b_2是布尔型的布尔表达式，则经逻辑运算后仍是布尔型；凡是谓词（或称原子）公式和时态公式均为布尔型。

3. 语句

时态逻辑中的每一条公式均可看成一时态语句。因此，那些具有特殊意义的谓词

gets, invar, halt, …

和量词符

$(\forall \cdots)$，$(\exists \cdots)$

构成的公式，就其时态语义来说，也可称它们为时态语句。执行这些语句相当于计算这些时态逻辑公式的值。如此，时态逻辑当作一种程序设计语言就成为现实了。

(1) 赋值语句。传统的程序设计语言中的赋值语句，在时态语义下，称时态赋值。因此，可用时态逻辑公式$\bigcirc z=e$来表示这一语句，称时态赋值语句。就逻辑意义言$\bigcirc z=e$在δ上为真当且仅当z在下一时刻取e值，即执行该语句等于计算该时态逻辑公式的值。形式定义

$$\bigcirc z=e \triangleq (\exists u)(\text{invar}(u) \wedge u=e \wedge \text{fin}(z=u))$$

其中fin是最后谓词，$\text{fin}(A)$表示A最后为真，这里是指执行完该赋值语句后，时态变量z在下一时刻取得e值。值得注意的是变量u不在e中自由出现。

(2) 联连语句。设A和B是两个时态公式，如果

维普资讯 http://www.cqvip.com

它们是毗连的，则用“；”把它们连起来，成为

$$A;\ B$$

称如此形式的结构为毗连时态语句。其逻辑意义是当前公式A操作完毕，紧接着操作后一公式B，也即该毗连结构在δ上是真当且仅当存在一个$i\leqslant|\delta|$，使得前A在$\langle\delta_0,\ \cdots,\ \delta_i\rangle$上是真，并且在$\delta$的$i$一截段$\delta^{(i)}=\langle\delta_i,\ \cdots,\ \delta|_{\delta}|\rangle$上，后$B$也真。

(3) 条件语句。时态结构

$$\textbf{if } A \textbf{ then } B \textbf{ else } C$$

称时态条件语句。它定义为

$$\textbf{if } A \textbf{ then } B \textbf{ else } C \triangleq (A\to B)\vee(\sim A\to C)$$

其逻辑意义是if A then B else C在δ上是真当且仅当在δ上A和B均真或者在δ上$\sim A$和C均真。

例如，计算两个数x和y的最大值并置于z中，则用时态语句描述是

$$\textbf{if } x\geqslant y \textbf{ then} \bigcirc z=x \textbf{ else } \bigcirc z=y$$

就其时态语义言，该公式在任何δ上为真，当且仅当在相应区间的最后状态上z等于x和y在初始状态上的值的最大者，所以

$$(\textbf{if } x\geqslant y \textbf{ then } \bigcirc z=x \textbf{ else } \bigcirc z=y)\leftrightarrow \bigcirc z=\max(x,\ y)$$

是有效的，即

$$\vdash (\textbf{if } x\geqslant y \textbf{ then } \bigcirc z=x \textbf{ else } \bigcirc z=y)\leftrightarrow \bigcirc z=\max(x,\ y)$$

(4) 循环语句。时态结构

$$\textbf{while } A \textbf{ do } B$$

其中A和B都是时态公式，称此为时态当量循环语句。它定义为

$$\textbf{while } A \textbf{ do } B \triangleq \Diamond(A\wedge B\wedge\bigcirc\sim A)\vee\Box(A\wedge B)$$

其逻辑意义是while A do B为真当且仅当$\Diamond(A\wedge B\wedge\bigcirc\sim A)\vee\Box(A\wedge B)$为真。又$\Diamond(A\wedge B\wedge\bigcirc\sim A)$是真意味着存在一个$t$，使得对$A$进行$t$次测试，每次均为真，故$B$被执行$t$次。当进行第$t+1$次测试时，结果为假，则停做$B$而执行其后继，在这种情况下，表明循环是可终止的；$\Box(A\wedge B)$为真意味着对任意的$t$，使得对$A$施行$t$次测试，每次均为真，由于$t$的任意性，故执行循环体有无限制的可能，在这种情况下，表明循环是不可终止的，其后继也未作任何具体规定。

例如，计算x和y的最大公约数，并将其置于z中，即$\bigcirc z=\gcd(x,\ y)$，现用时态当量循环语句描述如下：

$$\textbf{while}\sim(x=0)\textbf{ do}(\bigcirc z=x\wedge\bigcirc x=y \bmod x\wedge\bigcirc y=z)$$

另外，时态公式halt(A)的时态语句与传统程序设计语言中的stop和end语句具有相同意义。这样看来，时态逻辑直接用于程序设计语言就其内容言远比传统程序设计语言更为丰富，功能更强。下面我们继续定义时态公式的执行。

以时态公式定义的时态程序语句，其执行约定为从左到右，先内后外地进行。因此，语句

$$\bigcirc y=x+3\wedge\bigcirc x=0$$

是具有程序的语法错误。因为在计算$x+3$时，x的值未知，要改正，只需把二赋公式交换位置即可。

例如，input(x) $\bigcirc y=1\wedge\Box$output(y)$\wedge$ halt($x=0$)$\wedge$gets(x, $x-1$)$\wedge$gets(y, $2y$)。

执行这串时态公式（或称时态语句）按从左到右步骤是

①input(x)：比如从键盘上敲入4；

②$\bigcirc y=1$：给y赋初值1；

③$\Box$output(y)：意味着在δ的$\delta^{(1)}$状态下，输出1；

④halt($x=0$)：先测试x等于0？若为0，则停止执行，否则继续执行其右邻公式；

⑤gets(x, $x-1$)：将$x-1$的值，放回x中；

⑥gets(y, $2y$)：将y的2倍值贮于y中；

⑦检查δ的$\delta|_{\delta}|$是否$|\delta|=0$？若等于0，则停止执行，否则进入$\delta^{(2)}$状态。自该状态后，总是从$\Box$output(y)项开始重复，直至输出16后，$x=0$时停止。

三、举 例

现在用上述给出的时态逻辑语言来设计程序。这里用的两个例子，都是假定在给定了类似于Lisp解释程序系统下进行的。由此可见，使用时态逻辑语言书写程序，其结构尤为简洁和清晰。

例1 对于给定的非负整数集N上的输入x_2及整数集I上的输入x_1，试写出计算$z=x_1^{x_2}$的程序。

计算$z=x_1^{x_2}$的算法由框图（图1）给出。

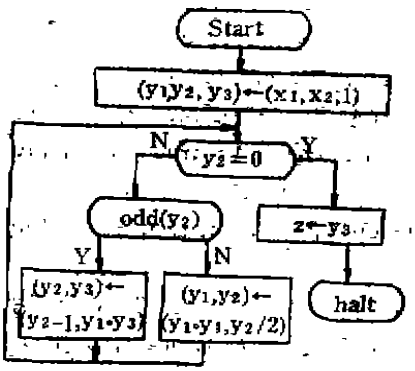

图1 计算$z=x_1^{x_2}$的程序框图

维普资讯 http://www.cqvip.com

相应于框图1的时态逻辑语言程序如下：

input(x_1, x_2)∧○$y_1=x_1$∧○$y_2=x_2$∧○$y_3=1$∧
(while~($y_2=0$)do
(if y_2 mod 2=0 then(○$y_1=y_1*y_1$∧○$y_2=y_2/2$∧○$y_3=y_3$)
else(○$y_1=y_1$∧○$y_2=y_2-1$
∧○$y_3=y_3*y_1$)))
○$z=y_3$∧□output(z)∧halt($y_2=0$)。

例2 试写一个判定整数集上$x>2$的任给定x是否是素数的程序。

寻找素数的算法由下面框图给出（图2）。

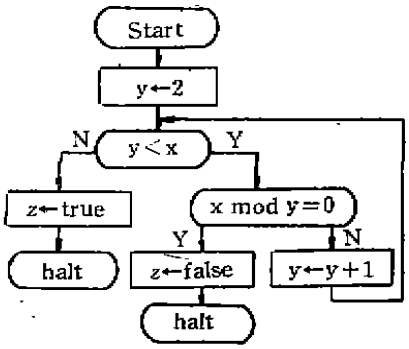

图2 判定x是否是素数的算法流程图

对框图2，同样可用时态语句来书写出具体的可执行的时态程序：

input(x)∧○$y=2$∧(while $y<x$ do
(if~(x mod $y=0$) then○$y=y+1$
else(○z=false∧□output(z)∧
halt( )))
∧○z=true∧□output(z)∧halt( )。

注意：halt( )中的( )，意味着空公式。

四、讨 论

用时态逻辑公式编写问题求解的程序是极有意义的。因为用时态语句写的时态程序，其正确性可直接通过证明程序中的每条语句的有效性得到，甚至对循环程序的可终止性，就某种意义而言也是可判定的，也就是说只要能从理论上证明循环条件谓词（或称时态公式）在给定的领域上不是在时态语义下普遍有效的，就可断定该循环终将是可终止的。另外，若能证明循环条件谓词蕴涵循环体，则循环程序总是部分正确的。

同Lisp和prolog一样，这种时态程序的可执行性是依赖于时态逻辑公式的解释系统。当然，要使这种时态逻辑程序语言完善起来，还要做许多工作。

参考文献

〔1〕 Ben Moszkowski "Executing Temporal Logic Programs" Computer Laboratory, University of Cambridge, England, 24 August 1984.

〔2〕 刘清"建立在时态逻辑公式演绎基础上的程序设计"，《计算机应用与软件》，1989，1。

（上接75页）

内容：数据库和知识系统的理论和实践的新发展，包括：复杂对象、并发控制、数据库机、数据模型、数据库程序设计语言和持久性程序设计、演绎数据库、不完备信息、知识表示和非单调推理、面向对象的数据库、性能评价，查询语言和优化、空间和瞬态数据，事务管理

（**江海** 辑自 Bull. EATCS, №.38, 1989年6月号）

3. **会议名称**：Fifth Annual IEEE Syposium on Logic in Computer Science

时间：1990年6月4—7日 **地点**：美国费城（Philadelphia）

程序委员会主席： Prof. John C. Michell

Department of computer Science, Stanford University, Stanford, CA 94305, USA

论文截止日期：1989年 12月1日

内 容：抽象数据类型，自动推导，并发性，构造性教学，数据库理论，有限模型理论，知识表示，λ和组合演算，计算复杂性的逻辑方面，人工智能中的逻辑，逻辑程序设计，模态和时间逻辑，程序逻辑和语义，重写规则，软件规格说明，类型系统，验证。（**江海** 辑）